IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: ITIKARLAPALLI *et al.*	Art Unit: 2168
Appl. No.: 10/709,522	Examiner: SANDERS, AARON J
Filed: 05/11/2004	Attorney Docket No.: ORCL-003/OID-2003-253-01
For: Simplifying Implementation of Custom Atomic Transactions in a Programming Environment	

Amendment and Response Under 37 C.F.R. §§ 1.116

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Final Office Action mailed 02/15/2007, Applicants submit the following amendments and remarks.

Amendments to the claims are reflected in the listing of claims which begin on page **02** of this paper.

Remarks begin at page number **05** of this paper.

It is not believed that extensions of time or fees for net addition of claims are required, beyond those which may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, then such extensions of time are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required therefor (including fees for net addition of claims) are hereby authorized to be charged to Deposit Account No.: 20-0674.

Listing of Claims

Claim 1 (Previously Presented): A method of implementing an atomic transaction using a program logic, said method comprising:

requesting in said program logic a transaction identifier for said atomic transaction;

generating said transaction identifier in a transaction manager in response to said requesting;

specifying in said program logic a plurality of combinations for execution in a sequential order, wherein each of said plurality of combinations contains said transaction identifier, a task procedure, and a rollback procedure, wherein said task procedure implements a part of said atomic transaction and said rollback procedure is designed to rollback said task procedure;

executing said task procedures in said sequential order;

keeping track of said rollback procedures in said transaction manager; and

executing said rollback procedures in a reverse order of said sequential order if said atomic transaction is to be aborted, wherein said rollback procedures are identified according to said keeping.

Claim 2 (Original): The method of claim 1, wherein said transaction identifier is unique to each of the atomic transactions.

Claim 3 (Previously Presented): The method of claim 1, wherein said keeping comprises storing data representing said rollback procedures in a stack.

Claim 4 (Original): The method of claim 3, wherein said stack is stored in a memory.

Claim 5 (Original): The method of claim 1, further comprising examining a status returned by execution of one of said task procedures and performing said aborting if said status indicates an error.

Claim 6 (Original): The method of claim 1, wherein said aborting is performed asynchronously.

Claims 7 (Previously Presented): A computer readable medium carrying one or more sequences of instructions representing a program logic for execution on a system, said program logic implementing an atomic transaction, wherein execution of said one or more sequences of instructions by one or more processors contained in said system causes said one or more processors to perform the actions of:

requesting an identifier for said atomic transaction;

setting a variable to equal said identifier;

specifying a plurality of combinations for execution, wherein each of said plurality of combinations contains said transaction identifier, a task procedure, and a rollback procedure, wherein said task procedure implements a part of said atomic transaction and said rollback procedure is designed to rollback said task procedure; and

aborting said atomic transaction by specifying said identifier associated with an abort procedure to cause said rollback procedures to be executed.

Claim 8 (Original): The computer readable medium of claim 7, wherein said specifying comprises including each of said plurality of combinations in a single procedure call.

Claim 9 (Original): The computer readable medium of claim 7, further comprising examining a status returned by execution of one of said task procedures and performing said aborting if said status indicates an error.

Claims 10 - 15 (Canceled)

Claim 16 (Previously Presented): A computer system comprising: a memory storing a plurality of instructions; and a processing unit coupled to said memory and executing said plurality of instructions to support implementation of an atomic transaction in a programming environment, said processing unit being operable to:

request in a program logic a transaction identifier for said atomic transaction;

generate said transaction identifier in a transaction manager in response to said requesting;

specify in said program logic a plurality of combinations for execution in a sequential order, wherein each of said plurality of combinations contains said transaction identifier, a task procedure, and a rollback procedure, wherein said task procedure implements a part of said atomic transaction and said rollback procedure is designed to rollback said task procedure;

execute said task procedures in said sequential order;

keep track of said rollback procedures in said transaction manager; and

execute said rollback procedures in a reverse order of said sequential order if said atomic transaction is to be aborted, wherein said rollback procedures are identified according to said keeping.

Claim 17 (Original): The computer system of claim 16, wherein said transaction identifier is unique to each of the atomic transactions.

Claim 18 (Previously Presented): The computer system of claim 16, wherein said processing unit is operable to store data representing said rollback procedures in a stack to perform said keep.

Claim 19 (Original): The computer system of claim 18, wherein said stack is stored in a memory.

Claim 20 (Original): The computer system of claim 16, wherein said processing unit is further operable to examine a status returned by execution of one of said task procedures and to perform said aborting if said status indicates an error.

Claim 21 (Previously Presented): The computer system of claim 16, wherein said processing unit is operable to execute said rollback procedures asynchronously.

REMARKS

Claims 1-21 were examined in the Final office action mailed on 02/15/2007 (hereafter "First Final Office Action". All claims were finally rejected under 35 U.S.C. § 102 (b) as being anticipated by United States Patent Application 5,701,480 naming as inventor Raz (hereafter "Raz").

By virtue of this amendment, claims 10-15 are sought to be canceled.

Applicant respectfully traverses the outstanding final rejections with respect to the remaining pending claims for reasons explained below.

For example, independent claim 1 recites:

> A method of implementing an atomic transaction using a program logic, said method comprising:
>
> requesting in said program logic a transaction identifier for said atomic transaction;
>
> generating said transaction identifier in a transaction manager in response to said requesting;
>
> specifying ***in said program logic*** a plurality of combinations for execution in a sequential order, ***wherein each of said plurality of combinations contains said transaction identifier, a task procedure, and a rollback procedure***, ***wherein said task procedure implements a part of said atomic transaction*** and said rollback procedure is designed to rollback said task procedure;
>
> executing said task procedures in said sequential order;
>
> keeping track of said rollback procedures in said transaction manager; and
>
> ***executing said rollback procedures in a reverse order of said sequential order if said atomic transaction is to be aborted***, wherein said rollback procedures are identified according to said keeping.
>
> (Previously presented claim 1, ***Emphasis Added***)

Based on the above, the Examiner's attention is directed to the below specific features of previously presented claim 1:

1. The task procedures and rollback procedures are specified in the program logic; and

2. The rollback procedures are executed in a reverse order of the order in which the task procedures are executed.

By providing a programmer the ability to specify task procedures and corresponding rollback procedures, enhanced control is provided to the programmer. The programmer can specify any desired logic in the rollback procedures.

Raz does not disclose or reasonably suggest either of the features noted above.

While Raz also addresses rollback as pointed by the Examiner, the technique there is different from the invention of claim 1.

In particular, resource manager 91 of Figure 5A of Raz provides the same rollback logic to every application program 90, and thus it is Applicant's position that the rollback procedures are not specified by the claimed program logic.

In support of this position, Applicants point to some relevant portions of Raz:

> Turning now to FIG. 5A, there is shown a block diagram of the programming and data structures used in the digital computer 20 of FIG. 1. for scheduling transactions and enforcing global transaction commitment ordering. ***Global and local transactions are initiated, for example, by application programs 90. To commit the results of transactions to state memory 28, 29 and to recover from failures, the digital computer is provided with a resource manager (RM) 91 that, for example, performs the operations shown in FIG. 3.*** The resource manager 91, for example, also manages a transaction list (TL) 93 as further described below with reference to FIG. 6. ***In general, a resource manager (RM) is a software component that manages state memory resources affected by committing transactions in such a way that the memory state of the resources can be restored before the transaction is committed by effectively undoing all of the changes introduced by the transaction***. (Col. 18, Lines 14-30 of Raz, ***Emphasis Added***)

From the above, it may be readily appreciated that resource manager 91, shared by all application programs 90, provides the rollback facility (using the technique of Figure 3 of Raz). Thus, the claimed rollback procedures are not specified in the program logic (or application programs of Raz).

Even if such rollback procedures are deemed to be specified by the application program of Raz, it is Applicant's position that there is no disclosure or suggestion in Raz that the rollback procedures are executed in the reverse order of execution of the task procedures (forming the atomic transaction).

In support of this position, Applicants point the Examiner to the relevant portions of Raz:

> To enable the recovery of memory records after a partial system failure, it is necessary for the application program to keep backup copies of the records in nonvolatile memory. When the computing system is restarted, ***the memory records to be recovered are replaced with the backup copies***. (Col. 2, Lines 1-6 of Raz, ***Emphasis Added***)
>
> To deal with the problem of ***possible failure*** when writing to non-volatile memory, there has been established a method of programming called "transaction processing" which guarantees that a portion of the non-volatile memory (referred to hereinafter as "state memory") will either be unaffected by a transaction or will be properly updated by results of a transaction, in the presence of the failures. ***Transaction processing is based upon the technique of making a back-up copy of state memory before the results of a transaction are written to state memory, and also writing in non-volatile memory an indication of either a first processing phase in which the back-up copy is being made, or a second processing phase in which the results of a transaction are being written to state memory, in order to indicate which copy might have been corrupted during a failure.*** For making a back-up copy of state memory, for example, the non-volatile memory 23 includes two banks of state memory 28 and 29. To provide an indication of which bank of stat memory might have been corrupted by a failure, the non-volatile memory 23 includes a memory location 30 for storing a switch or flag. (Col. 12, Lines 14-34 of Raz, ***Emphasis Added***)
>
> Turning now to FIG. 2A, there is shown a flow chart of a procedure for guaranteeing that when recovering from a failure, the state memory of the computer 20 shown in FIG. 1 is either unaffected by a transaction or is properly updated by the result of a transaction. Assume, for example, that the computer system is turned on after a power failure. In a first step 51, the central processing unit 21 reads the value of the switch 30 stored in the non-volatile memory 23. This switch indicates ***which of the two banks of state memory 28, 29 might possibly have been corrupted by the power failure. In step 52, the central processing unit 21 references the value of the switch to read the bank of state memory known not to have been corrupted, and to make a "working copy" of the data in the other bank of state memory***. Therefore, after step 52, both bank 28 and bank 29 of state memory have the same contents. (Col. 12, Lines 45-60 of Raz, ***Emphasis Added***)

From the above teachings, it is concluded that Raz relies on having a copy of the data prior to start of an atomic transaction, and uses the prior copy to rollback to a state prior to the start of execution of the atomic transaction in case of abort.

Due to such a technique, Raz would not have the need to execute rollback procedures (specified by the program logic) in addition to not having the motivation to execute the rollback procedures in the reverse order of execution of corresponding task procedures.

In rejecting claim 1, the Examiner has also quoted the below portions of Raz:

> A multi-versioning mechanism is employed in the "Rdb/VMS" (Trademark) and "VAX/DBMS" (Trademark) operating systems sold by Digital Equipment Corporation of Maynard, Mass. A "snapshot" mechanism eliminates the need for read locks and also prevents the blocking of read-only transactions by write locks. The "snapshot" mechanism permits a transaction to obtain, at any time, a consistent version of data existing at the time that the transaction begins. Write locks, however, are placed on records to be accessed by a read-write transaction, and the write locks are not released until the results of the read-write transaction are committed. Recoverability is further ensured by flushing to an "undo" log the "before-images" of records to be updated, and then flushing the updated records to state memory just before a transaction is committed. ***If a crash occurs, the updated records are replaced with "before images" that are obtained from the "undo log" to undo the effects of the failed transactions***. (Lines 39-45 of Col. 5 of Raz, ***Emphasis Added***)

The above quoted text of Raz suffers from the same deficiencies noted above. In particular, one of the "before-images" appears to be used to undo the effects of failed transactions.

This logic appears the same for all application programs and not dependent on what a programmer specifies in the application programs of Raz.

Thus, it is Applicant's position that Raz does not teach or suggest several features of independent claim 1 noted above. Accordingly claim 1 is believed to be allowable over Raz. Dependent claims 2-6 are also believed to be allowable at least as depending from an allowable base claim 1.

Independent claim 7 is also allowable over Raz at least in reciting in relevant portions:

> A computer readable medium carrying ***one or more sequences of instructions representing a program logic*** for execution on a system, said program logic implementing an atomic transaction, wherein ***execution of said one or more sequences of instructions*** by one or more processors contained in said system causes said one or more processors to ***perform the actions of***:

> ...
>
> ***specifying*** a plurality of combinations for execution, wherein each of said plurality of ***combinations contains*** said transaction identifier, ***a task procedure, and a rollback procedure***, wherein said task procedure implements a part of said atomic transaction and said rollback procedure is designed to rollback said task procedure; and
>
> aborting said atomic transaction by specifying said identifier associated with an abort procedure to cause said rollback procedures to be executed.
> (Previously Presented Claim 7, ***Emphasis Added***)

From the above, the elements of claim 7 are all performed by execution of instructions representing a program logic.

The program logic is akin to application programs 90 of Raz.

With such a construction, it is asserted that Raz does not teach or reasonably suggest the following features of claim 7:

1. The task procedures and rollback procedures are specified in the program logic; and

2. The program logic aborts the transaction by specifying the identifier of the transaction.

The basis for assertion 1 has already been explained with respect to claim 1 above.

Raz appears to contemplate recovery from crashes in the middle of transactions due to system errors and power failures, and there is no teaching or suggestion that the application programs 90 have any role in such recovery.

In particular, all the recovery of Raz appears to be performed only within resource manager 91 (or other components external to application programs 90), which is different from the application programs 90.

As such, Raz does not teach or suggest the features of independent claim 7 at least for some of the reasons noted above. Thus, independent claim 7 is believed to be allowable over the art of record. Dependent claims 8-9 are also believed to be allowable at least as depending from an allowable base claim.

Similarly, independent claim 16 recites in relevant portions:

> ....
>
> specify in said ***program logic*** a plurality of combinations for execution in a sequential order, wherein each of said plurality of ***combinations contains*** said ***transaction identifier, a task procedure, and a rollback procedure***, wherein said task procedure implements a part of said atomic transaction and said rollback procedure is designed to rollback said task procedure;
>
> execute said task procedures in said sequential order;
>
> keep track of said rollback procedures in said transaction manager; and
>
> ***execute said rollback procedures in a reverse order of said sequential order*** if said atomic transaction is to be aborted, wherein said rollback procedures are identified according to said keeping.
>
> (Previously presented independent claim 16, ***Emphasis Added***)

Raz does not appear to teach at least the highlighted features of independent claim 16 for reasons similar to those given above. Thus, independent claim 16 is believed to be allowable. Dependent claims 17-21 are also believed to be allowable at least as depending from an allowable base claim.

Conclusion

Thus, it is believed that all rejections have been overcome. The Examiner is respectfully requested to withdraw the final rejection and continue examination. The Examiner is invited to telephone the undersigned representative at 707.356.4172 if it is believed that an interview might be useful for any reason.

Respectfully submitted,

/Narendra Reddy Thappeta/
Signature
Printed Name: Narendra Reddy Thappeta
Attorney for Applicant
Registration Number: 41,416

Date: April 16, 2007